REMARKS

This Amendment is in response to the Office Action mailed October 11, 2006.

At the outset, the Applicants' representative wishes to thank Examiner Patel for his time, professionalism and courtesy during the recent telephone interview of April 10, 2007.

In the Office Action, claims 20-23, 71-79, 81, 94-97 are rejected under 35 U.S.C. §102(e) as being anticipated by Rabin et al. Reconsideration and withdrawal of these rejections are respectfully requested. Claims 1-19, 24-25, 82-90 are rejected under 35 U.S.C. §103(a) as being unpatentable over Rabin et al. in view of Hall et al. Reconsideration and withdrawal of these rejections are respectfully requested.

Independent claim 1

Rabin et al. identifies "an instance of a software" as follows:

> **Using this invention, a software vendor may have a specific piece of software, such as a specific application program or a specific book or song, which the vendor wishes to sell or lease, or otherwise distribute in a controlled manner, to users. Each particular copy of the software which is intended to be installed on or used on a user's device, is referred to as an instance of that software, or as a software instance. (Col. 3, lines 29-36)**

and Rabin et al. tell us that there is one unique tag for each piece of software:

> **Embodiments of the invention also encompass a tag server that accepts a copy of specific vendor software and produces a plurality of tags, one tag per instance of the software, with each tag uniquely identifying an instance of software with which it is associated. Each tag preferably comprises at least one of the name of the software associated with the tag, a unique number of the instance of software associated with the tag, and hash function values computed on portions of the instance of software associated with the tag. A digital signature mechanism may be used to digitally sign the tags and to securely transmit the tags to an intended receiver, such as a user device or guardian center or to the software vendor.**
> **Methods encompassed by the invention include a method for supervising usage of software. The method includes the steps of creating an instance of software and creating a tag that is uniquely associated with the instance of software. The method then distributes the instance of software and securely distributes the tag to a user device and receives the instance of software and the associated tag at the user device. The method then detects**

an attempt to use the instance of the software on the user device and determines if the attempt to use the instance of the software is allowable by determining a status of the tag that is associated with the instance of software to be used.

In the method, tag creation includes steps of assigning a unique number to the instance of software and computing a first hash function value on portions of the content of the instance of software. Then computing a second hash function value for the instance of software, the second hash function value combining the name of the software, the unique number of the instance of software, and the first hash function value. Next, the method includes the step of computing a tag that is uniquely associated with the instance of software, the tag including the name of the software, the unique number of the instance of software and the second hash value. (Col. 10, lines 31-67)

In Rabin et al., the <u>same</u> piece of software located on <u>different</u> user machines <u>will not</u> receive the same tag. In fact, Rabin et al.'s tags are unique to the piece of software, to the instance of software, and is unique for a same piece of software across users. That is, Rabin et al. teach that different copies of the same piece of software installed on different user devices (what Rabin et al. call an "instance of software") are assigned different tags. In Rabin et al., therefore, each tag for each instance of software is unique – meaning that each different piece of software installed on different user devices has its own unique tag.

In contrast, claim 1 is amended herewith to recite that

each different executable software component ... is uniquely associated with a unique identifier and is signed with a separate and unique PKI certificate, the separate and unique PKI certificate being uniquely identified at least by the unique identifier, ~~and~~ wherein identical executable software components in different ones of the plurality of gaming machines of the network connected gaming system are associated with identical identifiers and are signed with identical PKI certificates<u> such that non-identical executable software components in different ones of the plurality of gaming machines are associated with separate and different identifiers and are signed with separate and different PKI certificates, and such that no two non-identical executable software components in different gaming machines are signed with a same PKI certificate</u>

Claim 1, as amended, requires that each different executable software component is associated with a separate and unique PKI certificate such that:

a) "identical executable software components in different ones of the plurality of gaming machines of the network connected gaming system are associated with identical identifiers and are signed with identical PKI certificates";

b) "non-identical executable software components in different ones of the plurality of gaming machines are associated with separate and different identifiers and are signed with separate and different PKI certificates"; and

c) "no two non-identical executable software components in different gaming machines are signed with a same PKI certificate"

Rabin et al. do not teach this and this deficiency is not remedied by the secondary reference to Hall et al. Hall et al. is relied on for his teaching of limiting executable software to an allowed target environment using a cryptographic key corresponding to the target environment. A combination of the two references would, therefore, teach or suggest to the person of ordinary skill in this art, a system wherein each software component is assigned a unique tag, wherein identical software components in different computers were assigned different tags (as taught by Rabin et al.), and wherein execution of software is further limited to a target environment by a cryptographic key generated from characteristics of the target environment (as taught by Hall et al.). However, the embodiment of claim 1 is neither taught nor suggested by such a combination. Wholly untaught and unsuggested by a Rabin et al.-Hall et al. combination is the claimed "identical executable software components in different ones of the plurality of gaming machines of the network connected gaming system are associated with identical identifiers and are signed with identical PKI certificates, such that non-identical executable software components in different ones of the plurality of gaming machines are associated with separate and different identifiers and are signed with separate and different PKI

certificates, and such that no two non-identical executable software components in different gaming machines are signed with a same PKI certificate."

Claim 1, therefore, distinguishes over the applied combination of references to Rabin et al. and Hall et al. Reconsideration and withdrawal of the rejections applied to claim 1 and its dependent claims are, therefore, respectfully requested.

Independent claim 17

Claim 17 has been amended in manner that is similar to the manner in which independent claim 1 was amended:

> **code signing each executable software component subject to receiving certification with its respective separate and unique PKI certificate, each respective PKI certificate being uniquely identified at least by a unique identifier that is uniquely associated with the executable software component such that identical executable software components in different ones of the plurality of gaming machines of the network connected gaming system are associated with identical identifiers and are code signed with identical PKI certificates, such that non-identical executable software components in different ones of the plurality of gaming machines are associated with separate and different identifiers and are code signed with separate and different PKI certificates and such that no two non-identical executable software components in different gaming machines are code signed with a same PKI certificate, and**

Therefore, claim 17 cannot be said to be anticipated by Rabin et al., for the same reasons as advanced above relative to claim 1. Reconsideration and withdrawal of the rejections applied to claim 17 and its dependent claims are, therefore, respectfully requested.

Independent claim 20

Independent claim 20, as amended, recites:

> **code signing each authorized software component with a PKI certificate such that identical authorized software components in different ones of the constituent computers of the gaming system are code signed with identical PKI certificates, such that non-identical authorized software components in different ones of the constituent computers are code signed with separate and different PKI certificates and such that no two non-**

identical authorized software components in different ones of the constituent gaming machines of the gaming systems are code signed with a same PKI certificate;

configuring a separate software restriction policy for each authorized software component in each of the constituent computers of the gaming system, and associating the configured separate software restriction policy with the PKI certificate with which the authorized software component was code signed;

enforcing the associated software restriction policy for each code signed authorized software component such that ~~the~~ each code signed authorized software component in each of the constituent computers of the gaming system must be authorized to run by its associated separate software restriction policy.

That is, each authorized software component of each of the computers of the gaming systems must be authorized to run by its own SRP (Software Restriction Policy) – there being one such SRP for each of the constituent authorized software components in each of the computers of the gaming system. Moreover, the claim requires that the configured SRP be associated with the PKI certificate with which the authorized software component was code signed. In addition, the recited PKI certificate is subject to the detailed recitations relating to uniqueness as claimed in amended claim 1. In support of the rejection of claim 20, the Examiner again cites the same passages in Rabin et al.; namely, Col. 26, lines 50-60, Col. 27, lines 30-44, Col. 28, lines 5-15, Col. 28, Table 1, line 30 to Col. 30, line 20, and Col. 52 line 60 to Col. 53, line 25.

None of these passages teach or suggest this subject matter. Indeed:

Col. 26, lines 50-60 defines what is an "instance of software" is "Any individual copy of a specific software, such as, for example, a copy of a specific application program or a specific book or video, will hereafter be referred to as an instance of software or a software instance." Note that in Rabin et al., each software instance receives a different tag. That is, different instances of the same piece of software will receive different tags.

Col. 27, lines 30-44: teaches that a tag is generated for each instance of software. Rabin et al. explicitly tells us: "Typically, all instances of a specific software are identical. Preferably, a single tag is uniquely associated with a single instance of software produced by the software vendor."

Col. 28, lines 5-15: This passage tells us that THE supervising program in the user device verifies the validity of the instance of software requested by the user or by the device. Therefore, a single supervising program (NOT one for each software instance, as the Office Action states) verifies the validity of the tags of each instance of software in the user device.

Col. 28, Table 1, line 30 to Col. 30, line 20: This table is a glossary of terms. Nothing more.

Col. 52 line 60 to Col. 53, line 25: This passage details how the supervisory program SP 209 executes a call-up procedure to verify the tag of the instance of software requested by the user device or the user. Note, there is but a single supervisor program per user device, and not one SP209 for each software instance, as intimated by the Office.

The Examiner, in the telephone interview of April 10, 2007, also pointed to Col. 15, lines 12-31 and Col. 22, lines 42-60, as teaching that the supervisor program may include multiple policies. However, even if Rabin et al. were to teach multiple policies, Rabin et al. do not teach a) code signing with a unique and separate PKI certificate as claimed, nor any steps of

> ...**associating the configured separate software restriction policy with the PKI certificate with which the authorized software component was code signed;**
>
> enforcing the <u>associated</u> software restriction policy for each <u>code signed</u> authorized software component such that ~~the~~ each <u>code signed</u> authorized software component in each of the constituent computers of the gaming system must be authorized to run by its associated separate software restriction policy.

Rabin et al. do NOT teach the claimed code signing step, the associating step or the enforcing step – even if the separate supervisor program (SP 209) includes multiple policies.

The Office then states:

> **Rabin creates a PKI certificate specifically with unique software call-up policy for each unique hardware ID. It is clearly that the gaming system must be authorized to run by its associated separate software restriction policy (Col 28 lines 30-65)**

At the outset, Rabin et al. do no such thing. There are precisely four instances of the term "certificate" in Rabin et al. (and none are related to Rabin et al.'s method), and all describe the use of "Certificates of Authenticity", which are different animals altogether:

> **There are many types of software locking mechanisms. For example, a manufacturer can encrypt portions of a software program with the unique key. A customer who purchases the software is given the key which allows decryption and execution of the software. An example of such a software protection mechanism is a "Certificate of Authenticity" supplied with the purchase of software programs such as Microsoft Windows 98, manufactured by the Microsoft Corporation of Redmond, Wash. Microsoft and Windows98 are trademarks of the Microsoft Corporation. The Certificate of Authenticity indicates a unique product number. During installation of the software, the product number is requested by the software application and must be entered correctly by the user. If the product number entered matches a number expected by the application, the copy of the application is assumed to be legitimate and is allowed to be installed and executed as normal. If the number entered is incorrect, the software will not install properly.**
> **...**
> **Characteristics of Prior Art Systems**
> **Prior art techniques for protecting the unauthorized use of software and information suffer from a variety of problems. Systems which use a certificate of authenticity or key suffer in that one key allows unlimited usage of the program and nothing prevents copying of the key. As such, the owner of a copy of the software can pass his key or certificate along with the software or information to someone else who can use the certificate or key to install and run the software or to access the information. If one key allows only a single usage or a one-time execution, the problem of copying may be solved but then each usage requires a separate key to be entered. To be commercially acceptable most programs require multiple uses.**

Moreover, Rabin et al. do not create a "PKI certificate specifically with unique software call-up policy for each unique Hardware ID", as urged by the Office. That statement is simply untrue and unsupported by Rabin et al.

Page 31 of 44

Rabin et al. do state that:

> That is, if a pirate attempts to circumvent the usage supervision protection of the invention by duplicating the entire disk information and transferring the duplicated disk to another device, the invention can allow hardware device identification mechanisms to be incorporated into tag information and during tag validation (i.e. during call-up processing--to be explained), the hardware identification information can be checked accordingly. (Col. 43, lines 5-12).

There is no teaching in Rabin et al. of a) a separate policy for each user ID or b) a PKI certificate for each policy or for each hardware ID. In Rabin et al., the single supervisor program 209 of the user device performs the call up to check the validity of the unique tag for the requested software instance. Kindly recall that Rabin et al.'s tags are unique across all devices, even when the tags are associated with identical software instances within different devices – which is diametrically opposite to the claimed embodiment, in which "identical authorized software components in different ones of the constituent computers of the gaming system are code signed with identical PKI certificates."

Therefore, it is not believed that the 35 U.S.C. §102(e) rejection applied to claim 20 and to its dependent claims is untenable. Reconsideration and withdrawal of the rejections applied to claim 20 and its dependent claims are, therefore, respectfully requested.

Independent claim 22

As amended, claim 22 recites:

> **code signing each authorized software component with a PKI certificate such that identical authorized software components in different ones of the constituent computers are code signed with identical PKI certificates, such that non-identical authorized software components in different ones of the constituent computers are code signed with separate and different PKI certificates and such that no two non-identical authorized software components in different ones of the constituent gaming machines are code signed with a same PKI certificate;**
> configuring a path software restriction policy to prevent unauthorized software components from executing;

> configuring a path software restriction policy to prevent non-explicitly
> authorized software components from executing;
>
> enforcing the certificate software restriction policy configured for each
> of the code signed authorized executable software components of each of
> the constituent computers of the gaming system, and
>
> enforcing the path software restriction policies.

The comments immediately above relative to claim 20 are equally applicable here and, in the interest of avoiding duplicative arguments, the arguments advanced relative to claim 20 are repeated herein by reference. Reconsideration and withdrawal of the 35 U.S.C. §102(e) rejection applied to claim 22 and to its dependent claims are, therefore, respectfully requested.

Independent claim 24

As amended herewith, claim 24 recites:

> producing a separate and unique PKI certificate for each of the plurality
> of executable software ~~component~~ components within the gaming system
> subject to receive certification, each respective PKI certificate being
> associated with a unique identifier that is uniquely associated with the
> executable software component such that identical executable software
> components in different ones of the plurality of gaming machines of the
> network connected gaming system are associated with identical identifiers
> and are code signed with identical PKI certificates, such that non-identical
> executable software components in different ones of the plurality of gaming
> machines are code signed with separate and different PKI certificates and
> such that no two non-identical executable software components in different
> gaming machines are code signed with a same PKI certificate;

To anticipate this claim, Rabin et al. must teach that "a separate and unique PKI certificate is produced for each executable software component" ... "such that non-identical executable software components in different ones of the plurality of gaming machines are code signed with separate and different PKI certificates and such that no two non-identical executable software components in different gaming machines are code signed with a same PKI certificate."

In Rabin et al., no two tags are identical, as each uniquely identifies an instance of the software with which it is associated. This unique tag (again, no two of them are the same in

Rabin et al.) is uniquely associated with the instance of the software and is just that – unique. According to Rabin et al., identical executable software components on different machines would be associated with different tags – exactly the opposite of the claimed invention.

In contradistinction, claim 24 calls for identical authorized executable software components on different machines be code signed with identical PKI certificates, which could never happen in Rabin et al.

Claim 24, therefore, distinguishes over the applied reference to Rabin et al. Reconsideration and withdrawal of the rejections applied to claim 24 and its dependent claims are, therefore, respectfully requested.

Independent claim 25

Claim 25, as amended, recites:

> code signing each authorized executable software component with a separate PKI certificate that is unique to the authorized software component such that identical executable software components in different ones of the plurality of gaming machines of the network connected gaming system are code signed with identical PKI certificates, such that non-identical authorized software components in different ones of the plurality of gaming machines are code signed with separate and different PKI certificates and such that no two non-identical authorized software components in different gaming machines are code signed with a same PKI certificate;

The arguments advanced relative to claim 24 above are equally applicable here. These arguments, therefore, are incorporated herein by reference.

Independent claim 71

Claim 71 recites a step of:

> executing the code signed installation package upon every startup of any of the constituent computers of the gaming system or upon a command, wherein execution of any authorized executable file is predicated upon successfully executing the code signed installation package into which the authorized executable file is packaged.

The Patent Office points to Col. 47, line 56 to Col. 48, line 33 and Col. 52, line 60 to Col. 53, line 25 as teaching such a step. These passages are sufficiently short as to be reproduced herein in their entirety:

> In other embodiments, there may be call-up policies (CALL-UP_POLICY_SW) associated with individual instances of software 111-114. In this case, step 272 can examine the rules or tests of the call-up policy (CALL-UP_POLICY_SW) associated with the software content SW or the instance of software (INST_SW) 111-114 that was requested access by a user 213 in step 270. In another embodiment, if the user 213 of a user device 104 attempts to use an untagged instance of software, step 272 may mandate that a call-up is needed. In another embodiment, if the user 213 of a user device 104 uses tagged software for the first time, then step 272 may mandate that a call-up is needed. In another embodiment, the maximum allowed interval between successive call-up procedures is preferably determined by a combination of elapsed time in a user device 104, the number and duration of uses to instances of software 111-114, the number of times the device 104 is powered on, and/or by any other measure that is related to time or use of the device 104.
>
> Call-up processing will be discussed in more detail later. Essentially however, during call-up processing, the supervising program (SP) 209 in a user device 104 securely transfers a copy of the tag table 210 and the fingerprint table 126 to the guardian center 103 (FIG. 2). After verification, the guardian center 103 (FIG. 2) compares each tag TAG_INST_SWn in the tag table 210 against a list of compromised tags. The guardian center 103 (FIG. 2) can detect tags that are invalid or compromised in some manner.
>
> A usage supervision policy POLICY(TAG_INST_SW) associated with each tag can also be checked at the guardian center 103 (FIG. 2) to ensure that tags 120 (and therefore instances of software associated with the tags) are being used in compliance with the usage supervision policy POLICY(TAG_INST_SW). The policy may be for an entire user device 104-107 or on a per user 213 or per tag 120 basis. Also, for untagged software, the fingerprint table 126 can be compared against a fingerprint data structure (explained later) in the guardian center 103 (FIG. 2) to detect uses of infringing software INF_SW. After analysis of the tag table 210 and fingerprint table 126 are complete, the guardian center 103 (FIG. 2) prepares and sends a continuation message (CM) 212 (FIG. 2) back to the user device 104.

and

> A call-up is made in accordance with the CALL-UP_POLICY or CALL-UP_POLICY(TAG_INST SW) as explained above in response to a user's attempt to use an instance of software 111-114 on a user device 104-107. That is, when the user 213 attempts to use an instance of software 111-114 for which the time allowed before the next call-up according to the CALL-UP_POLICY of the user device 104 or the CALL-UP_POLICY(TAG_INST_SW) of the software (SW) for that instance has expired, the supervising program 209 on that device 104-107 initiates step

> **370.** In another embodiment, the SP 209 executes a call-up procedure at a chosen time before the expiration time, regardless of whether a use of an instance of software 111-114 is requested. The CALL-UP_POLICY can be maintained within the supervising program 209 on the user device 104. In addition, it is possible that a call-up may occur because a portion of the supervising program 209, executing regardless of use requests, determines that it is time to perform a call-up. For example, it may take place as the result of a certain number of BOOTUPS (power-ups) of a user device 104-107 having taken place or the first use of untagged software.
>
> If the call-up to the guardian center 103 (FIG. 2) in step 371 fails, then processing proceeds to step 376 where punitive action may be performed by the supervising program (SP) 209 on the user device 104. In the preferred embodiment, the supervising program (SP) 209 will perform a new call-up, retrying several times before beginning punitive action. In the case that punitive action is necessary in step 376, the punitive action may merely be to inform the user 213 that the instance of software 111-114 that was requested is temporarily inaccessible due to a communications failure.

As the Examiner will note from re-reading these passages, no teaching or suggestion is made therein regarding any "executing **the code signed installation package** upon every startup of any of the constituent computers of the gaming system or upon a command, wherein execution of any authorized executable file is predicated upon successfully executing the code signed installation package into which the authorized executable file is packaged", as required by claim 71. Indeed, **no code signed installation packages are taught in Rabin et al.**, nor is there any teaching or suggestion therein of executing such a code signed installation package (or any teaching of re-installing software components – which is what an installation package does - **upon every startup** of any of the constituent computers of the gaming system, as again required by claim 71.

Rabin et al. do teach that the supervisory program may execute a call up procedure to verify the tag of an instance of software (which is not the same as a code signed installation package) after a predetermined number of boot ups:

> In another embodiment, the SP 209 executes a call-up procedure at a chosen time before the expiration time, regardless of whether a use of an instance of software 111-114 is requested. The CALL-UP_POLICY can be maintained within the supervising program 209 on the user device 104. In

addition, it is possible that a call-up may occur because a portion of the supervising program 209, executing regardless of use requests, determines that it is time to perform a call-up. For example, it may take place as the result of a certain number of BOOTUPS (power-ups) of a user device 104-107 having taken place or the first use of untagged software. (Col. 53, lines 3-15).

However, such does not rise to the level of a teaching of <u>executing a code signed installation package</u> upon every startup of any of the constituent computers of the gaming system, as claimed herein. Recall that the tags in Rabin et al. are associated with instances of software, and not to code signed installation packages. In view of the foregoing, it is respectfully submitted that claim 71 is not anticipated by Rabin et al.

Reconsideration and withdrawal of the 35 U.S.C. §102(e) rejection of claim 71 over the Rabin et al. reference is, therefore, respectfully requested.

<u>Independent claim 73</u>

Claim 73 recites:

> **packaging the authorized executable files into a code signed installation package;**
> **configuring certificate rule policies to enable execution of the code signed installation package;**
> **configuring enforcement of the policies, and**
> **re-installing the code signed installation package at every startup of any of the constituent computers of the gaming system or upon a command, wherein execution of any authorized executable file is predicated upon successfully executing the code signed installation package into which the authorized executable file is packaged.**

The arguments advanced relative to claim 71 are repeated here, by reference. Claim 73 calls for re-installing a code signed installation package at every startup of the constituent computers of the gaming system. In contrast, Rabin et al. teach that a Supervisory Program (SP 209) <u>executes a call-up after a predetermined number of boot-ups</u> to verify the software instance's tag. Claim 73, however, requires that <u>a code signed installation package</u> ($\neq$ Rabin et

al.'s Supervisory Program 209 or tags) be re-installed <u>at every startup</u> of any of the constituent computers. By definition, therefore, Rabin et al. cannot anticipate claim 73. The Office will recall that even if a single recitation is not taught by the reference, the §102(e) rejection must fall. Reconsideration and withdrawal of the 35 USC §102(e) rejection of claim 73 over the Rabin et al. reference is, therefore, respectfully requested.

Independent claim 75

Independent claim 75 includes similar recitations as does claim 73:

> **packaging the at least one non-executable file into at least one code signed installation package;**
> **configuring certificate rule policies to enable execution of the at least one code signed installation package;**
> **configuring enforcement of the policies, and**
> **re-installing the at least one code signed installation package at every startup of any of the constituent computers of the gaming system or upon a command.**

Rabin et al. demonstrably do not teach re-installing the code signed installation packages(s) upon every startup, as claimed. The arguments advanced immediately above relative to claims 71 and 73 are incorporated herein by reference. Reconsideration and withdrawal of the 35 USC §102(e) rejection of claim 75 over the Rabin et al. reference is, therefore, respectfully requested.

Independent claim 77

Independent claim 77, similarly, recites:

> ...
> **re-installing the at least one code signed installation package at every startup of any of the constituent computers of the gaming system or upon a command.**

The arguments advanced above relative to claims 71, 73, and 75 are also applicable here, and are incorporate herein by reference, as if repeated here in full. Reconsideration and withdrawal of the rejections applied to claim 77 and its dependent claims are, therefore, respectfully requested.

Independent claim 79 and its dependent claims

Claim 79 recites:

> packaging at least one authorized non-executable file that ~~control~~ controls the scheduling of the at least one authorized executable software component into at least one code signed installation package, each of the at least one code signed installation packages including a predetermined PKI certificate;
> configuring certificate rule policies to enable execution of the at least one code signed installation package in selected ones of the plurality of gaming machines; and
> configuring enforcement of the certificate rule policies; and
> downloading the at least one code signed installation package into the selected ones of the plurality of gaming machines;
> executing the at least one code signed installation package.

It is respectfully submitted that Rabin et al. do not teach packaging one or more authorized non-executable files that control the scheduling of the at least one authorized executable software component into at least one code signed installation package, as claimed – and much less that each of the code signed installation packages includes a predetermined PKI certificate, as also claimed.

The Examiner repeatedly (and in support of the rejection of this claim also) cites Col. 26, lines 50-60, Col. 27, lines 30-44, Col. 28, lines 5-15, Col. 28, Table 1, line 30 to Col. 30, line 20, and Col. 52, line 60 to Col. 53, line 25 as teaching the claimed subject matter. However, none of these passages teach packaging one or more authorized non-executable files that control the scheduling of the authorized executable software components <u>as required by the claim</u>. This

subject matter simply is not present in Rabin et al. <u>If the Office maintains its position that such claimed subject matter is indeed taught by Rabin et al., the Office is respectfully requested to specifically point out where such teachings of packaging one or more authorized non-executable files that control the scheduling of the authorized executable software components are to be found in this reference</u>. Failing a teaching of this recitation, an anticipation rejection cannot stand.

Reconsideration and withdrawal of the anticipatory rejections applied to claim 79 and to its dependent claims are, therefore, respectfully requested.

Independent claim 82 and its dependent claims

Independent claim 82, as amended herewith, recites:

> **code-signing means for enabling the manufacturer or subcontractor to associate a separate and unique PKI certificate with each authorized software component subject to regulatory certification such <u>that</u> identical authorized software components subject to regulatory certification in different ones of the plurality of gaming machines of the network connected gaming system are code signed with identical PKI certificates<u>, such that non-identical executable software components in different ones of the plurality of gaming machines are code signed with separate and different PKI certificates, and such that no two non-identical executable software components in different gaming machines are code signed with a same PKI certificate</u>.**

Therefore, to anticipate this claim, Rabin et al. must teach code-signing means ... to associate a separate and unique PKI certificate with each authorized software component *such that identical authorized software components subject to regulatory certification in different ones of the plurality of gaming machines of the network connected gaming system are code signed with identical PKI certificates, such that non-identical executable software components in different ones of the plurality of gaming machines are code signed with separate and different PKI certificates, and such that no two non-identical executable software components in different*

gaming machines are code signed with a same PKI certificate, as claimed. In Rabin, each instance of software is assigned a unique tag, as detailed in Col. 10, lines 31-67:

> **Embodiments of the invention also encompass a tag server that accepts a copy of specific vendor software and produces a plurality of tags, one tag per instance of the software, with each tag uniquely identifying an instance of software with which it is associated.**
>
> **...**
>
> **In the method, tag creation includes steps of assigning a unique number to the instance of software and computing a first hash function value on portions of the content of the instance of software. Then computing a second hash function value for the instance of software, the second hash function value combining the name of the software, the unique number of the instance of software, and the first hash function value. Next, the method includes the step of computing a tag that is uniquely associated with the instance of software, the tag including the name of the software, the unique number of the instance of software and the second hash value.**

That is, in Rabin et al., no two tags are identical, as each uniquely identifies an <u>instance</u> of the software with which it is associated. This unique tag (no two of them are the same in Rabin et al.) is constructed by assigning a unique number, computing a first hash function and a second hash function, after which the tag is computed using the name of the software, the unique number and the second hash value. This tag that is uniquely associated with the instance of the software is just that – unique.

In contradistinction, claim 82 calls for identical authorized software components subject to regulatory certification in different ones of the plurality of gaming machines of the network connected gaming system to be code signed with <u>identical</u> PKI certificates, which could never happen in Rabin et al.

Claim 82, therefore, distinguishes over the applied reference to Rabin et al. Reconsideration and withdrawal of the rejections applied to claim 82 and its dependent claims are, therefore, respectfully requested.

Independent claim 94

Independent claim 94 recites:

> **generating a unique code signed PKI certificate for a predetermined software module of each authorized game;**
>
> **generating an executable companion file for each authorized game, wherein the executable companion file is configured to execute faster than the authorized game;**
>
> **code signing both the predetermined software module and its executable companion file with the generated PKI certificate;**
>
> **enforcing software restriction policy rules for preventing non-authorized software components from executing;**
>
> **enforcing software restriction policy rules for enabling execution of selected ones of the authorized games;**
>
> **attempting to execute each executable companion file, and**
>
> **adding only those games to the menu of authorized games whose executable companion file has not been denied execution by the software restriction policy rules.**

The claim requires that an executable companion file be generated for each authorized game – and that both the predetermined software module and this generated executable companion file be code signed with the generated PKI certificate. Rabin et al. do not teach any such executable companion file, and much less an executable companion file that has been code signed with the same PKI certificate as a software module of the game.

The claim also requires that the executable companion file execute faster than the authorized game, which is something that Rabin et al. do not teach at all. Moreover, Rabin et al. do not teach any step of attempting to execute any companion file and adding only those games to the menu of authorized games whose executable companion file has not been denied execution by software restriction policies. Please note that if Rabin et al. do not teach an executable companion file, withdrawal of the anticipatory rejection is not discretionary, it is mandatory.

Note paragraphs [0122] to [0124] in the originally-filed specification for an explanation of the claimed "companion file":

[0122] Fig. 20 shows a companion Hello component, according to another aspect of the present invention. Reference numeral 2000 in Fig. 20 illustrates a method to generate a code signed companion software component. Each game comprises an aggregate of executable and non-executable software components, usually comprising files such as *.exe, *.dll, *.dat, *.xml. In general, all the software components are dependent of one component named the main program or the game entry. Starting the execution of the main game component is a lengthy process, as a large number of dependent executable components and graphics need to be verified (SRP verification) and started. ...
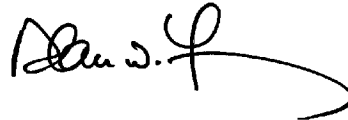
[0123] Another embodiment of the present invention, therefore, provides a method to quickly verify the policy enforcement on a game without starting the entire game, in order to generate the list of available games to be made available to the player in a menu. For each game, a very short companion .dll file may be created having, for example, only one line of code « Return "HELLO" » which would return the exemplary "HELLO" string when called. Assuming "Infinity.dll" 2010 is the main game component file name 2002 (or friendly name), then the companion file may be named "Infinity.Hello.dll" 2018. ...

[0124] It is to be noted that code signing two distinct software executables with the same certificate is a deviation from the method taught earlier in this document. However, the fact that the role of the companion file is very well defined, as having for example only one line of code « Return "HELLO" » which would return the "HELLO" string when called, this does not present an issue with the regulators or the certification lab.

Rabin et al. do not teach anything of the sort. Moreover, none of the passages identified by the Examiner even remotely teach any type of code-signed executable companion file, the successful execution of which is a necessary predicate to adding a game to a menu of authorized games, as required by claim 94. It is respectfully submitted that Rabin et al. simply do not teach any such subject matter. It is respectfully submitted that the Office is not at liberty to simply ignore claim language for which it finds no counterpart in the applied art, as is the case with claim 94 and the recited "executable companion file" in the Office Action of 10/11/06. Reconsideration and withdrawal of the rejections applied to claim 94 and its dependent claims are, therefore, respectfully requested.

Applicants' attorney believes that the present application is now in condition for an early

allowance and passage to issue. If any unresolved issues remain, the Examiner is respectfully

invited to contact the undersigned attorney of record at the telephone number indicated below, and

whatever is required will be done at once.

Respectfully submitted,

Date: April 11, 2007   By:

Alan W. Young
Attorney for Applicants
Registration No. 37,970

YOUNG LAW FIRM, P.C.
4370 Alpine Rd., Ste. 106
Portola Valley, CA 94028
Tel.: (650) 851-7210
Fax: (650) 851-7232

\\Ylfserver\ylf\CLIENTS\JMG\CYBS\5858 (Trusted Game Download)\5858 AMEND.4.doc